Introduction to Web 2.0
Programming
Using JavaScript, CSS, and jQuery

By John Valance
JValance Consulting
Email: jvalance@sprynet.com

Vermont Midrange Users Group Conference – Sept. 28, 2010
About John Valance

- Independent consultant
  - Specialty is helping iSeries shops develop web applications
  - Training, mentoring, consultation and coding
- Over 20 years iSeries/AS400 experience
- Ten years of web development experience
- Frequent presenter on web development topics
- Web scripting language of choice = PHP
  - Also experienced with SQL, Java, RPGLE
- Trainer for Zend Technologies
  - Teaches Intro to PHP for RPG programmers
What We Will Cover

- Web 2.0 application characteristics
- Some background – Legacy web apps
- CSS
- JavaScript
- DOM
  - Legacy and W3C
- jQuery framework
- Ajax with jQuery
- Web 2.0 application demo
- Corvette America dealer order entry demo
What Do We Mean By Web 2.0?

- Hard to Define
- Includes a broad array of paradigms for interacting through the web, including:
  - social networking
  - multi-media
  - web services
  - mobile devices
- We will focus on the enabling technologies used in browser-based applications
  - Introduce the concepts and techniques
  - Native browser technologies only
    - Not Flash or other plug-in related techniques
Characteristics of Web 2.0 Applications

- Highly Interactive / High performance
  - Response time and functionality rival PC desktop applications

- Very dynamic interface
  - expand/collapse
  - drag/drop
  - intellisense

- UI is controlled by Javascript and CSS
  - Lots of client-side code
Characteristics of Web 2.0 Applications

- Interaction with Server via Ajax
  - Used to retrieve bits of data from the server without reloading the entire page
  - Can also update the server through Ajax calls

- Use of Javascript frameworks to simplify client-side code and cross-browser issues
  - jQuery
  - Prototype
  - Dojo
  - ExtJS
  - script.aculo.us
Background: HTTP and Static Web Content

- HTTP = Request / Response protocol
  - Client (browser) requests a file from server
  - Server responds with the file requested
- Content is Static
  - HTML documents and other files are returned “as-is”
  - No pre-processing step prior to response
- Updates to content are done offline by HTML coder
  - HTML is hard-coded
Legacy Web (1.0) Application

We can create database applications using a simple HTML front-end

- Still uses HTTP request/response protocol
- Can deliver dynamic content via server-side scripts
  - e.g.: PHP, JSP, CGI, etc.
- Dynamic content typically retrieved from a database
- Database content can also be updated via HTML form inputs
- Interactions with server typically involve a form submission as the request, with a complete HTML document returned as the response
Some Aspects of Web 1.0 Applications

- Limited functionality
- Limited types of inputs and UI controls
- Slow response (due to large payload of HTML documents + ancillary files)
- Page reloads may reposition at top of page

But…

- You *can* create useful applications with simple HTML forms + server scripts.
Languages Involved in a Typical Web Application

Client Side:
- HTML
- CSS
- JavaScript

Server Side:
- SQL
- `<your favorite server scripting language>`:
  - PHP
  - Java
  - RPG
  - PHP calling RPG
Legacy Web Application Demo

Customer Inquiry Application

- Uses HTML form submission on client
- Uses PHP and mySql on server

“Self-submitting” form script
- Calls itself
  - If customer number present on request, it will retrieve database info
  - Else it will just present input form
Modern Web 2.0
Application Architecture

Heavy reliance on JavaScript and CSS

- CSS determines visual aspects of HTML elements
  - Colors, fonts, visibility, positioning, borders…

- JavaScript is used to:
  - React to events (mouse movements and clicks, keystrokes, page load…)
  - Modify CSS and other attributes of HTML elements
  - Add and remove HTML elements or change text contents
  - Call scripts on server via Ajax
    - retrieve or update database information
What is CSS?

- CSS stands for Cascading Style Sheets
- Styles define how to display HTML elements
  - Styles were added to HTML 4.0 to help separate content from presentation.
- Styles are normally stored in Style Sheets
  - a list of presentation attributes that together define the style of a page or an entire site.
- External Style Sheets are stored in .css files
  - External Style Sheets can save you a lot of work, since site-wide style changes can be made in one file.
CSS Syntax

selector {
    property: value;
    property: value;
    ...
}

- **Selector**: identifies a part of the document to be styled
  - HTML tag name, Class name, or a Unique ID (more on this coming…)

- **Property**: A specific presentation attribute to be styled
  - color, font-weight, border attributes, visibility

- **Value**: How the presentation attribute should be styled
  color: red; font-weight: bold; border: 2px solid blue;
Where can Styles be specified?

- Inside a single HTML element
  ```html
  <table style="border:none; color:blue">
  </table>
  ```

- Inside the `<head>` element of an HTML page
  ```html
  <head>
    <style type="text/css">
      table { border:none; color:blue }
    </style>
  </head>
  ```

- In an external CSS file
  ```html
  <head>
    <link rel="stylesheet" type="text/css" href="siteStyle.css" />
  </head>
  ```
Examples of CSS Selectors

- HTML Tag Name:
  BODY { font: arial; font-size: 12pt; color: navy }  
  - Can use any HTML tag name  
  - Applies to all occurrences of the tag throughout a document

- Class Name - precede with period (.):
  .error { color: red; font-weight: bold} <p class="error">Invalid email address</p>  
  - Can specify the same class on many different HTML tags

- Unique ID – precede with hash (#):
  #shipto { visibility: hidden }  
  <div id="shipto"> <table>... </div>
  - ID name should only occur once in HTML document
Demo of CSS Stylesheet

sitestyle.css, used in customer inquiry
demo
What is JavaScript?

- It isn’t Java, but similar syntax
  - C type syntax
- Scripting language for web browsers
  - Runs on the client-side only
  - All browsers have built-in JavaScript interpreter – you don’t buy it or install it.
- Interpreted at run-time (as page loads)
What Can JavaScript Do?

- Manipulate the HTML document after it has been sent to the browser in a myriad of ways
- Handle events
  - mouse clicks, cursor movement, etc.
- Communicate with user via pop-up alerts.
- Handle forms and input values
- Alter CSS attributes of HTML elements
- Add, change, delete HTML elements
- Open & close windows, and communicate between windows
- Read and write cookies
- Ajax – call server scripts without page reload
JavaScript Event Handling

- JavaScript can be used to react to events in the browser
  - mouse clicks, mouse movement, keystrokes, movement in and out of form fields, etc.
  - One of JavaScript’s most useful capabilities with regards to Web 2.0
- Can code “event handlers” as attributes of HTML tags
- Event handlers start with “on…”
  - Onclick, onfocus, onblur, onmouseover, onmouseout
- Event handler attributes specify some javascript code to execute as their value
  - Can be individual statements or function call

```html
<body onload="setInputDefaults();">  
<input name="company" onblur="this.value=this.value.touppercase();">  
<tr onmouseover="this.className='hilite';"> 
```
JavaScript Demo

- Simple form validation script
- Some dynamic HTML examples
DOM is how JavaScript interacts with HTML document elements.

HTML document is a hierarchy of elements defined by tags.

As browser loads document, these elements are loaded into objects that JavaScript can easily access.

JavaScript uses dot (.) notation to access objects and their properties:

```javascript
document.searchForm.searchField.value = 'Enter keyword';
```

Object properties can be read or modified by JavaScript.
- Some properties are read-only.
DOM Object Properties

Each DOM object can have properties:

- Tag attributes:
  
  `<img src="companylogo.gif">`
  `
  `<h1 style="color: blue; font-size: 14pt">`

- Tag contents:
  `
  <p>This is a short paragraph.</p>`

- Nested HTML tags (= nested JS objects):
  `
  <document>
  
  <form name="orderForm">
  
  <input name="zipCode">
  
  document.orderForm.zipCode
  
  </form>
  
  </document>`
Navigating the DOM

Legacy DOM (DOM Level 0, pre-IE4)

- Limited object model
- `document` object has properties that are arrays of specific HTML collections
  - `anchors[]`, `applets[]`, `forms[]`, `images[]`, `links[]`
  - `forms[]` object is the most important of these.
  - Each `forms[]` object also contains a collection of `elements[]`, which are the form’s input elements
- These arrays can be iterated over using a loop and an incremented index variable.
- Properties can by read and changed, but DOM level 0 does not allow altering document text content or re-flowing the document.
- Example:
  - `document.forms[0].elements[3].value = '123.45';`
Naming DOM Elements

- It is easier to navigate the DOM by naming elements

```html
<form name="myForm">
  <input type="text" name="UserName">
</form>

<script type="text/javascript">
  document.myForm.UserName = 'JOHNV';
</script>

- Can also use id attribute to get a specific element

```html
<input type="text" id="UserName">

<script type="text/javascript">
  var userName = document.getElementById('UserName');
  userName.value = 'JOHNV';
</script>
W3C DOM

- Implements a fully accessible and modifiable tree structure object model

```html
<html>
  <head>
    <title>This is a Document!</title>
  </head>
  <body>
    <h1>This is a header!</h1>
    <p id="excitingText">
      This is a paragraph! <em>This is emphasized</em>!
    </p>
    <p>This is also a paragraph, but it's not nearly as exciting as the last one.</p>
  </body>
</html>
```

From: Traversing the DOM by Mike West - http://dev.opera.com/articles/view/traversing-the-dom/
W3C DOM Tree

From: Traversing the DOM by Mike West - http://dev.opera.com/articles/view/traversing-the-dom/

Sept. 28, 2010 – VTMUG Conference.  Introduction to Web 2.0 – John Valance (Copyright 2010)
Navigating W3C DOM

- W3C DOM is implemented as a tree of various types of Node objects
- Each Node object defines properties and methods for traversing and manipulating the tree.
  - Properties
    - childNodes - a list of children of the node
    - firstChild, lastChild, nextSibling, previousSibling, parentNode
  - Methods
    - appendChild(), removeChild(), replaceChild(), insertBefore()
**Types of Nodes**

Every Node object has a `nodeType` property that specifies what kind of node it is.

<table>
<thead>
<tr>
<th>nodeType</th>
<th><code>nodeType</code> constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Node.ELEMENT_NODE</td>
</tr>
<tr>
<td>Text</td>
<td>Node.TEXT_NODE</td>
</tr>
<tr>
<td>Document</td>
<td>Node.DOCUMENT_NODE</td>
</tr>
<tr>
<td>Comment</td>
<td>Node.COMMENT_NODE</td>
</tr>
<tr>
<td>Attr</td>
<td>Node.ATTRIBUTE_NODE</td>
</tr>
</tbody>
</table>
Getting a Specific Node

- You could start at the document root, and work down to the desired element
  - Very cumbersome
  - Impossible if elements are added/deleted dynamically
- Best way is to assign an ID attribute
  - Use `document.getElementById('myTagId')`
- Can also get all elements by tag name
  - `document.getElementsByTagName('input')`
    - Returns an array of all input elements in the document
  - Can start at a specific node within document tree

```javascript
var itemsTable = document.getElementById("items");
var rows = itemsTable.getElementsByTagName("tr");
var numItems = rows.length;
```
Traversing the DOM examples

- **Starting at document root, without IDs**
  ```javascript
  var theHtmlNode = document.childNodes[0];
  var theBodyNode = theHtmlNode.childNodes[1];
  var theParagraphNode = theBodyNode.childNodes[1];
  alert( "theParagraphNode is a " +
          theParagraphNode.nodeName + " node!" );
  ```

- **Starting at a specific node, retrieved by ID**
  ```javascript
  var paragraphNode =
      document.getElementById('excitingText');
  alert( "The exciting text is " +
          paragraphNode.innerHTML);
  ```
Problems with Basic JavaScript

- Cross-browser compatibility issues
- Awkwardness and Complexity of Code
  - Abstract DOM traversal and selection
    - Arrays, Loops, Indexes
    - Referencing Nodes and Elements
    - Testing types and attributes of nodes
- Performance
jQuery to the Rescue

- Most widely used JavaScript framework
- Greatly simplifies coding of browser scripting
  - DOM selection and traversal
  - Dynamic UI controls
  - Ajax calls
- Seamless integration with CSS
  - Easy to select elements via JavaScript (why it’s called jQuery)
  - Easy to modify HTML attributes and CSS properties
- Intuitive syntax (well… somewhat)
- Open source and free (download from jQuery.org)
- High performance and small file size (~72K)
- No plug-ins required
  - Built on pure JavaScript
Example: Toggle Visibility

- Basic JavaScript
  ```javascript
  function toggleBox () {
    var elem = document.getElementById('box');
    if (elem.style.visibility == "hidden") {
      elem.style.visibility = "visible";
    } else {
      elem.style.visibility = "hidden";
    }
  }
  ```

- Using jQuery
  ```javascript
  $('#box').toggle();
  ```
jQuery Syntax

```
$("<selector>").<method>();
```

- $( )
  - This calls jQuery
  - Synonymous with calling `jQuery( )`
  - This example could be coded `jQuery("#box").toggle();`

- $("#box")
  - This selects the element with id="box"

- $("#box").toggle()
  - This toggles the visibility (hidden/visible) of the box element
    - Same as if statement in previous slide

- Could also use
  - `show()`, `hide()`, `slideDown()`, `slideUp()`,
  - `slideToggle()`, `fadeIn()`, `fadeOut()`
Selectors

- jQuery selectors = CSS selectors
  - HTML tag name
    - $(‘input’) – selects all <input> elements
    - $(‘p’) – selects all <p> elements
  - Class attribute
    - $(‘.inputLabel’) – selects any element with class="inputLabel"
  - ID attribute
    - $(‘#mainContent’) – selects the element with id="mainContent"
Combined Selectors

$(`p.helpText´)
- Select all paragraphs with class="helpText"
- No space between tag and class

$(`div#shippingInfo .label´)
- Select all elements with class="label", but only within the <div> tag with id="shippingInfo"
- Note the space between the two selectors, indicates descendent relationship

$(`h1, h2, h3´)
- Select all <h1>, <h2>, and <h3> tags
- Commas means a list of selectors
Methods

Once elements are selected, we can call a variety of methods to alter the document

- **hide()**, **show()** and related methods
  - Saw these earlier
- **addClass()**, **removeClass()**
  - Add and remove a CSS class
- **css()**
  - Directly alter values of CSS properties
- **attr()**, **removeAttr()**
  - Change or remove HTML tag attributes
- **text()**
  - Change text contained between open/close tags
- **insertBefore()**, **insertAfter**, **prepend()**, **append()**, **remove()**, **replaceWith()**, **clone()**
  - Allows insertion, deletion, moving, copying of HTML elements
Ajax

- Stands for Asynchronous JavaScript and XML
- Allows calling a server script from within an HTML page via JavaScript
  - Server script can be any language (PHP, Java, RPG)
  - Call is asynchronous (JavaScript doesn’t wait for response)
    - Can specify a JavaScript function to be called when response is received
- No page reload
- Can retrieve and update data on server very quickly
  - Sub-second response is typical (like a local PC application)
  - Can modify page contents or styling based on what’s returned by the server script
    - Use jQuery selectors and methods to update page content
- Ajax calls are simplified by using jQuery
  - Eliminates cross-browser syntactical differences
Making an Ajax Call

- $.get(<url>, <function-name>);
  function clickButton() {
    var url = 'getCustomer.php?custNum=1234';
    $.get(url, processResponse);
  }
  function processResponse( data ) {
    alert('data returned = ' + data);
  }

- Can also use $.post(<url>, <function-name>);
  - Depends on what the server script expects and how much data is being passed

- Data returned by server script can be any format
  - XML, HTML, JSON
JSON

- **JSON = JavaScript Object Notation**
- Simpler, more compact format than XML
  - Basically a list of name:value pairs enclosed in braces
    ```json
    {"error":true, "message":"Invalid zip code"}
    ```
- Easy to use and parse
- PHP has `json_encode()` function
  - Transform variables into JSON as
    ```json
    {<var-name>:<var-value>}
    ```
  - Can be used with arrays, including nested arrays
- **JavaScript can easily parse JSON into variables**
  - Can use jQuery functions or stand-alone parsers
Demo Corvette America

- Auto parts & accessories distributor in PA
- Created order entry application for their wholesale customers (dealers)
- 2-man RPG shop with no prior web development experience
  - Team also included a graphic designer from marketing department
- Wanted to create the application in-house, using PHP hosted on iSeries (Zend Core)

Features
  - Sophisticated item search
  - Multiple shopping baskets per customer
  - Ajax calls for item validation and adding items to baskets
Summary

- Web 2.0 is the current standard for web application development
- Provides rich UI and high performance
  - On par with desktop applications
- Capabilities based on JavaScript and CSS, applied dynamically to HTML documents
- More client-side coding
- Ajax technology allows JavaScript to communicate directly with server
  - Data retrieved can be inserted into HTML dynamically
- jQuery (and other JS frameworks) can greatly simplify and standardize coding of Web 2.0 applications
Inquiries

Please contact John Valance at:

jvalance@sprynet.com
802-355-4024

Happy coding!